Поволжский государственный университет телекоммуникаций и информатики

Программы вокруг нас

Предпрофильная подготовка школьников
9-х классов

Автор: к.т.н. Назаренко Петр Александрович
доцент кафедры информационных систем и технологий ПГУТИ

Программы вокруг нас

Мы познакомимся с профессиями программиста, специалиста по информационным технологиям и родственными специальностями. Узнаем об особенностях работы людей, чей труд связан с программированием и информационными технологиями, о том, какие задачи они решают, о перспективах этих профессий. Получим представление о языках и технологиях программирования, о средствах разработки программ, их типах и разновидностях. На практике познакомимся с основами одного из самых современных и быстро развивающихся языков программирования – Python, научимся создавать на этом языке программы, простые и не очень, и даже такие, которые продемонстрируют нам, на чём основана работа Интернета.

Программное обеспечение и его виды

Системное:

- Операционные системы - Windows, UNIX, Linux, Android и др.

- Вспомогательные программы (дополнительное системное ПО – *утилиты:* просмотр каталогов, файлов, печать, сравнение файлов и т.п.)

Виды программного обеспечения

Прикладное:

- Текстовые редакторы (Word и т.п.)
- Графические редакторы (PhotoShop)
- Математические программы (MATLAB, MathCAD, Mathematica)
- Системы управления базами данных
- Программы для работы с Интернетом (браузеры)
- Программы профессионального назначения

Виды программного обеспечения

Программы профессионального назначения:

- Автоматизированные системы управления технологическими процессами (АСУТП, SCADA)
- Системы автоматизированного проектирования (САПР, CAD) самого разного назначения - для машиностроения, электроники, антенн и т.д.

Области применения программного обеспечения

- Коммерческая деятельность (*программы общего назначения, базы данных, банковские программы, интернет-программирование*).
- Промышленная деятельность и транспорт (*SCADA-системы, автоматизированное проектирование, связь*).
- Научные исследования (*моделирование процессов, анализ данных*).

Области применения программного обеспечения

- Образование (*обучающие программы, математические вычисления* – *MathCAD, MATLAB, Mathematica*).

- Медицина (*диагностика, хирургические роботы*)

- Развлечения (*игры, воспроизведение музыки и видео и т.п.*).

- Искусственный интеллект (*в самых разных проявлениях*).

Интернет-программирование

Применение:

- Сайты фирм, организаций и учреждений
- Интернет-магазины
- Прочие

Характеристики профессий

Специальности:

- Программист
- Системный программист
- Руководитель разработки программного обеспечения
- Администратор баз данных
- Специалист по информационным системам
- Системный администратор информационно-коммуникационных систем

Характеристики профессий

Должности:

- Младший программист (техник-программист)
- Программист
- Старший программист (инженер-программист)
- Ведущий программист

Характеристики профессий

Выполняемые работы:

- Разработка и отладка исходных текстов программ
- Проверка работоспособности и улучшение (оптимизация) исходных текстов
- Объединение различных программных модулей и окончательная проверка работы готовых программ
- Разработка требований к программам и проектирование программ

Характеристики профессий

Разработка и отладка программ

- Формулировка решаемой задачи в терминах математики и определение этапов решения задачи (*алгоритмизация*)
- Написание текста программы (*кода*) с использованием языка программирования
- Оформление текста программы по заданным требованиям
- Проверка работы программы и отладка её исходного текста (устранение ошибок)

Характеристики профессий

Новые профессии («Атлас новых профессий»):

- *Архитектор информационных систем*
- *Дизайнер интерфейсов*
- Разработчик моделей «больших данных» (Big Data)
- ИТ-аудитор
- Кибертехник умных сред

Реально существующие:

- Специалист по работе с данными (Data Scientist)

Языки программирования

Считается, что существует более 8000 самых разных языков программирования и постоянно появляются новые.

Очерёдность появления:

https://ru.wikipedia.org/wiki/Хронология_языков_программирования

Языки программирования

По принадлежности к виду (*парадигме*) программирования:

- императивное
  - процедурное (Паскаль, Си)
  - объектно-оиентированное (Си++, C#, Java, Python)
- декларативное
  - функциональное (LISP, Erlang и др.)
  - логическое (Prolog)

Языки программирования

По способу получения работающей программы:

- интерпретирующие (Python, PHP и др.),
- компилирующие (Паскаль, Си, Си++, C#).

Новые языки:
Rust, Kotlin (предполагается на смену Java), Swift
https://ru.wikipedia.org/wiki/Сравнение_языков_программирования

Процесс создания программы

1. Набор исходного текста программы (*кода*) на одном из языков программирования. Используется почти любой текстовый редактор, например, «Блокнот» или Notepad++. Результат – файл с этим кодом.

2. Из исходного кода получают либо объектный файл (*компиляция*), либо сразу исполняемые команды в памяти машины (*интерпретация*). При компиляции используется программа-компилятор.

Процесс создания программы

3. При компиляции из объектного файла (модуля) или набора таких модулей получают исполняемую двоичную программу (для операционных систем Windows – EXE-файл). Используется программа-компоновщик (она же – редактор связей).

Исходный текст
→ объектный модуль
→ исполняемый файл

prog1.pas → prog1.obj → prog1.exe

Процесс создания программы

При использовании интерпретирующего языка программирования исполняемый файл не создаётся, а для работы программы нужен сам интерпретатор.

Именно таким языком является Python.

Инструментальные средства

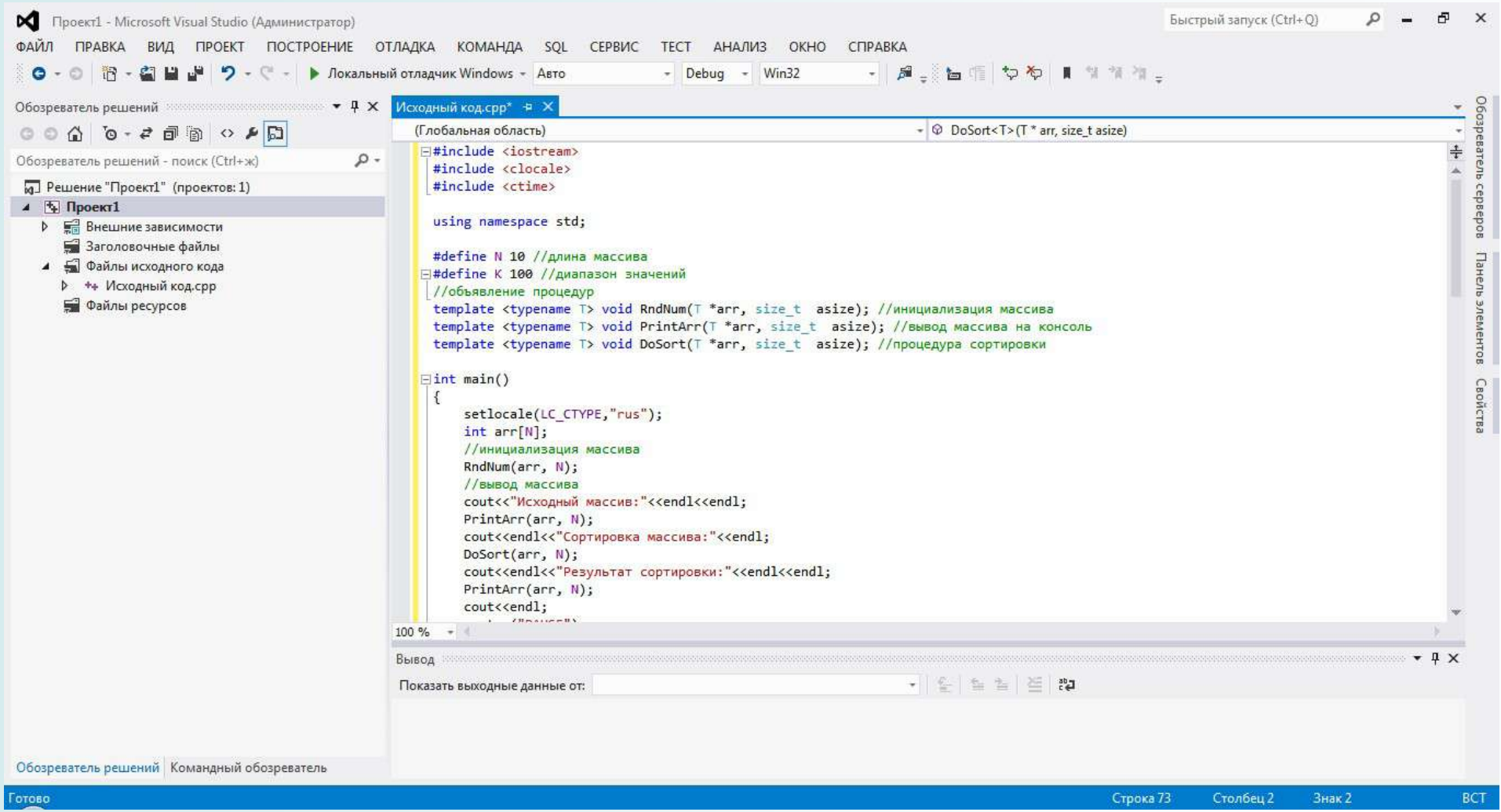
Интегрированная среда разработки

IDE
(Integrated
  Development
  Environment)

MS Visual Studio, Qt, Delphi, C++ Builder, Code::Blocks, Eclipse, NetBeans, Lazarus, wxDev-C++

Инструментальные средства

Microsoft Visual Studio 2012



Программирование на языке Python

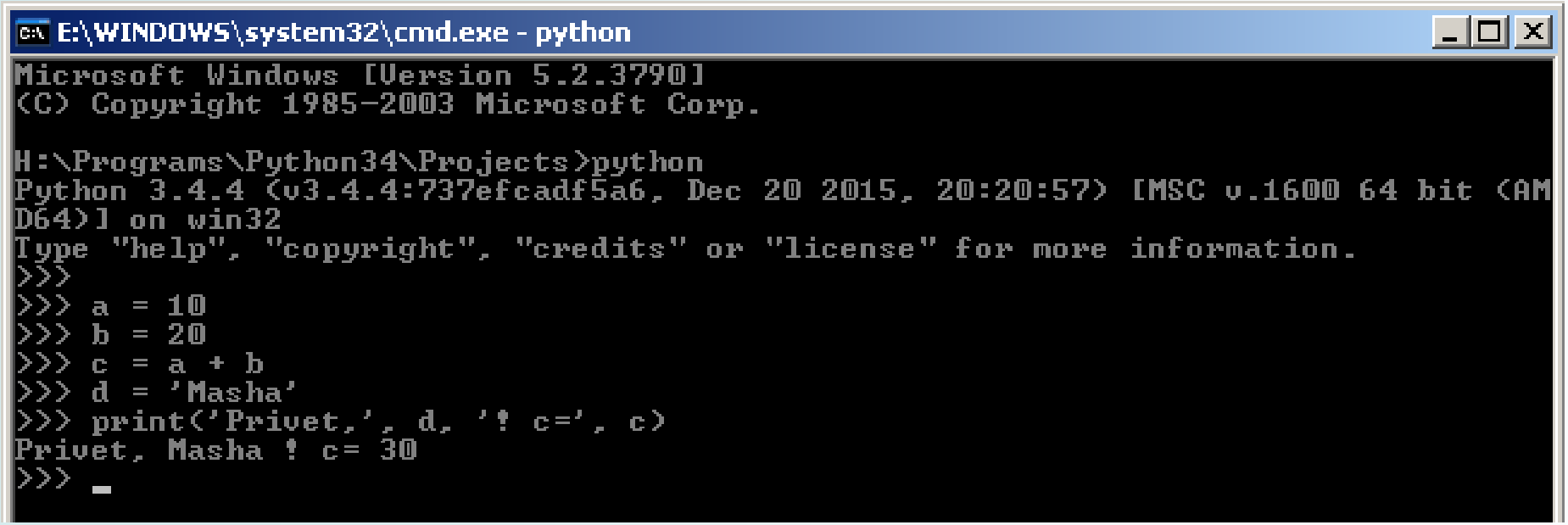
Лучше использовать новую версию Python 3 (Python 3.4.4 – 3.7.4)

Кодировка – UTF-8

Программирование на языке Python

Написание программы

Собственно «среда разработки» Python:



Программирование на языке Python

Написание программы

Редактор Notepad++

```
1
2  a = 10
3  b = 20
4  c = a + b
5  d = 'Masha'
6  print('Privet,',d,'! c=',c)
7  input('Для выхода нажмите <Enter>.')
8
```

Основные структуры программирования

Линейное выполнение команд

```
1 a = int(input('Введите значение переменной a: '))
2 b = int(input('Введите значение переменной b: '))
3 c = a + b
4 print('c =',c)
5 d = int(input('Введите значение переменной d: '))
6 e = c/d
7 print('e =',e)
8 input('Для выхода нажмите <Enter>.')
9
```

Основные структуры программирования

Упражнения

1. Запустите программу несколько раз с разными данными (целыми числами).
2. Удалите из строк №1 и 2 слово int или int и ближайшую к нему пару скобок ( ). Запустите программу. Сравните результат с предыдущими запусками.
3. Восстановите исходную программу. Замените в строке №5 слово int на float. Введите переменную d как дробное число.

Основные структуры программирования

Ветвление

```
1  a = int(input('Введите значение переменной a: '))
2  b = int(input('Введите значение переменной b: '))
3  if a>b:
4      print('a больше b')
5      max=a
6  else:
7      if a<b:
8          print('a меньше b')
9          max=b
10     else:
11         print('a и b равны')
12         max=0
13 print('c =',max)
14 input('Для выхода нажмите <Enter>.')
```

Основные структуры программирования

Упражнения

1. Запустите программу несколько раз с разными данными (целыми числами), так, чтобы проверить все варианты сравнения на больше, меньше и равно.

2. Проверьте результаты выполнения программы в каждом запуске.

Основные структуры программирования

Цикл

```
1  spisok = []  # Пустой список
2  N = int(input('Введите количество данных: '))
3  for i in range(N):  # Цикл ввода данных
4      a = int(input('Введите число: '))
5      spisok.append(a)
6
7  print('Содержимое списка: ',spisok)
8
9  a = int(input('Введите ещё одно число: '))
10 spisok.append(a)
11
12 for i in range(len(spisok)):  # Цикл вывода данных
13     print('Элемент №',i,'=',spisok[i],end='')
14     print()
15
16 i=0
17 while i<len(spisok):  # Ещё один цикл
18     spisok[i] += 1  # spisok[i] = spisok[i] + 1
19     i += 1          # i = i + 1
20
21 print('Список после модификации: ',spisok)
22 input('Для выхода нажмите <Enter>.')
```

Графическое отображение данных

Изменим предыдущую программу:

```
1  from tkinter import *  # Подключение библиотек
2  import math as m
3
4  root=Tk()
5  windowHeight=root.winfo_screenheight()
6  windowWidth=root.winfo_screenwidth()
7
8  y0=windowHeight-100
9
10 root.title('График введённых данных')
11 root.geometry('{}x{}'.format(windowWidth,windowHeight))
12 root.resizable(True,True)
13 root.state('zoomed')
14
15 drawPanel=Canvas(root,bg='white',width=windowWidth,height=windowHeight)
16 drawPanel.pack()
17
18 spisok1 = []  # Пустой список
19 N = int(input('Введите количество данных: '))
20 for i in range(N):  # Цикл ввода данных
21     a = int(input('Введите число: '))
22     spisok1.append(a)
```

Графическое отображение данных

Продолжение:

```
23  print('Содержимое списка: ',spisok1)
24  spisok2 = spisok1.copy()  # Второй список
25  a = int(input('Введите ещё одно число: '))
26  spisok2.append(a)
27
28  for i in range(len(spisok1)):  # Цикл вывода данных
29    print('Элемент №',i,'=',spisok1[i],end='')
30    print()
31
32  i=0
33  while i<len(spisok2):  # Ещё один цикл
34    spisok2[i] += 1  # spisok[i] = spisok[i] + 1
35    i += 1           # i = i + 1
36
37  print('Список после модификации: ',spisok2)
38
39  Sc=10  # Масштабирующий коэффициент
40  def draw():  # Функция
41    x1=0
42    y1=y0
43    drawPanel.create_line(x1,y0,windowWidth,y0,fill='green')
44    for i in range(len(spisok1)):
```

Графическое отображение данных

Окончание:

```
44    for i in range(len(spisok1)):
45      x2=(i+1)*Sc
46      y2=y0-spisok1[i]*Sc
47      drawPanel.create_line(x1,y1,x2,y2,fill='red')
48      x1,y1=x2,y2
49
50    x1=0
51    y1=y0
52    for i in range(len(spisok2)):
53      x2=(i+1)*Sc
54      y2=y0-spisok2[i]*Sc
55      drawPanel.create_line(x1,y1,x2,y2,fill='blue')
56      x1,y1=x2,y2
57  # Конец функции
58
59  menu1=Menu()
60  menu1.add_command(label='Draw',command=draw)
61
62  root.bind('<Escape>',exit)
63  root.bind('<q>',exit)
64  root.config(menu=menu1)
65  root.mainloop()
```

Графическое отображение данных

Упражнения

1. Запустите программу несколько раз с разными данными (целыми числами).
2. Удалите оператор x1=0 из строки №50. Запустите программу. Сравните результат с предыдущими запусками.
3. Восстановите исходную программу. Измените значение масштабирующего коэффициента в строке №39 с 10 на 20 (или любое приблизительно похожее). Запустите программу.

Графический интерфейс пользователя

Программа эхо-сервера

```
import asyncio

class EchoServerClientProtocol(asyncio.Protocol):
    def connection_made(self, transport):
        peername = transport.get_extra_info('peername')
        print('Connection from {}'.format(peername))
        self.transport = transport

    def data_received(self, data):
        message = data.decode()
        print('Data received: {!r}'.format(message))

        print('Send: {!r}'.format(message))
        self.transport.write(data)

        print('Close the client socket')
        self.transport.close()

loop = asyncio.get_event_loop()
# Each client connection will create a new protocol instance
coro = loop.create_server(EchoServerClientProtocol, '127.0.0.1', 8888) # Loop-back interface - only demo mode
server = loop.run_until_complete(coro)

# Serve requests until Ctrl+C is pressed
print('Serving on {}'.format(server.sockets[0].getsockname()))
try:
    loop.run_forever()
except KeyboardInterrupt:
    pass

# Close the server
server.close()
loop.run_until_complete(server.wait_closed())
loop.close()
```

Программа эхо-сервера (1)

```
2   import asyncio
3
4   class EchoServerClientProtocol(asyncio.Protocol):
5     def connection_made(self, transport):
6       peername = transport.get_extra_info('peername')
7       print('Connection from {}'.format(peername))
8       self.transport = transport
9
10    def data_received(self, data):
11      message = data.decode()
12      print('Data received: {!r}'.format(message))
13
14      print('Send: {!r}'.format(message))
15      self.transport.write(data)
16
17      print('Close the client socket')
18      self.transport.close()
```

Программа эхо-сервера (2)

```
19
20  loop = asyncio.get_event_loop()
21  # Each client connection will create a new protocol instance
22  coro = loop.create_server(EchoServerClientProtocol, '127.0.0.1', 8888)
23  server = loop.run_until_complete(coro)
24
25  # Serve requests until Ctrl+C is pressed
26  print('Serving on {}'.format(server.sockets[0].getsockname()))
27  try:
28      loop.run_forever()
29  except KeyboardInterrupt:
30      pass
31
32  # Close the server
33  server.close()
34  loop.run_until_complete(server.wait_closed())
35  loop.close()
```

Программа эхо-клиента

```
1  import asyncio
2
3  class EchoClientProtocol(asyncio.Protocol):
4      def __init__(self, message, loop):
5          self.message = message
6          self.loop = loop
7
8      def connection_made(self, transport):
9          transport.write(self.message.encode())
10         print('Data sent: {!r}'.format(self.message))
11
12     def data_received(self, data):
13         print('Data received: {!r}'.format(data.decode()))
14
15     def connection_lost(self, exc):
16         print('The server closed the connection')
17         print('Stop the event loop')
18         self.loop.stop()
19
20 loop = asyncio.get_event_loop()
21 message = 'Hello World!'
22 coro = loop.create_connection(lambda: EchoClientProtocol(message, loop),'127.0.0.1', 8888)
23 loop.run_until_complete(coro)
24 loop.run_forever()
25 loop.close()
```

Где почитать

Стандарты на профессии:

- Программист
- Системный программист
- Руководитель разработки программного обеспечения
- Администратор баз данных
- Специалист по информационным системам
- Системный администратор информационно-коммуникационных систем

Где почитать

Python:

- https://python.org/
- Справочник по языку Python (на русском)
- Справочник по языку Python и библиотеке Tkinter (на русском)

Приложения

Запуск программы на языке Python:

Если на компьютере транслятор языка Python установлен полностью правильно (и только одна его версия) (обычно это бывает на личном компьютере), то для запуска программы из командной строки потребуется только указать её имя – klient22.py (и в некоторых случаях можно даже без расширения, например, klient22).

Если транслятор установлен правильно, но с некоторыми ограничениями, то нужно указать имя файла транслятора и полное имя программы (с расширением .py), например, python klient22.py Этот вариант является наиболее общим.

Приложения

Запуск программы на языке Python:

Если на компьютере установлены 2 версии транслятора языка Python - 2-я и 3-я, и требуется запустить программу для версии 3, а предыдущие способы запуска не помогают, то нужно:

1). разместить программу в каком-либо доступном на запись каталоге, например "Загрузки" (Downloads);

2). узнать путь к транслятору нужной версии (например, по ярлыку в меню "Пуск", свойствам этого ярлыка и указываемого им файла), скопировать этот путь в буфер обмена (клавишами Ctrl+C);

2а). Например, в Windows 10:

Меню "Пуск" -> Python 3.4 (command line) (или 3.6 или 3.7) -> (правой кнопкой мышки) "Дополнительно" -> "Перейти к расположению файла" -> Python 3.x (command line) -> Свойства -> "Объект" - скопировать всё содержимое.

3). перейти в выбранный доступный каталог (Downloads), если это не было сделано раньше;

4). выполнить в этом каталоге команду cmd и ввести путь (или вставить скопированный, Ctrl+V), при необходимости дописать в конце имя транслятора (python), и указать имя запускаемой программы (должно быть похожим на C:\Users\111\Downloads>C:\Users\111\AppData\Local\Programs\Python\Python37-32\python klient22.py).

Приложения

Дополнительные вопросы:

Для проверки работы сервера и клиента они запускаются в отдельных окнах (т.е. отдельными командами cmd).

В зависимости от того, в каком редакторе набиралась программа, может потребоваться указать правильную кодировку. Для этого в самом начале программы ставится

```
# coding: cp1251
```

или

```
# coding: utf8
```